Trust Method for Multi-Agent Consensus

Dariusz G. Mikulski [*a,c], Frank L. Lewis [°b], Edward Y. Gu [c], Greg R. Hudas [a]

[a] Ground Vehicle Robotics (GVR), U.S. Army RDECOM-TARDEC, Warren, MI, USA
[b] Automation & Robotics Research Institute, University of Texas at Arlington, Fort Worth, TX, USA
[c] Department of Electrical and Computer Engineering, Oakland University, Rochester, MI, USA

ABSTRACT

The consensus problem in multi-agent systems often assumes that all agents are equally trustworthy to seek agreement. But for multi-agent military applications – particularly those that deal with sensor fusion or multi-robot formation control – this assumption may create the potential for compromised network security or poor cooperative performance. As such, we present a trust-based solution for the discrete-time multi-agent consensus problem and prove its asymptotic convergence in strongly connected digraphs. The novelty of the paper is a new trust algorithm called RoboTrust, which is used to calculate trustworthiness in agents using observations and statistical inferences from various historical perspectives. The performance of RoboTrust is evaluated within the trust-based consensus protocol under different conditions of tolerance and confirmation.

Keywords: Trust Model, Multi-Agent Systems, Consensus, Convergence

1. INTRODUCTION

The next generation of advanced military robotic systems will be required to operate and co-operate in highly dynamic, unstructured, and hostile environments, such as urban warzones, natural or man-made disaster areas, and subterranean caves and mines[1]. These systems will also increasingly become more autonomous and more common in military operations, creating a need for robust strategic and tactical reasoning AI. In particular, military robots will need to decide to what capacity they will interact with other robots and humans, given the presence of uncertainty and partial information[2]. Cooperative multi-robot teaming applications will also need to handle general task coupling and communication-delay challenges for these interactions. Currently, there is no working theory that takes into account all of these issues[3].

Generally speaking, an artificial agent is supplied with simple rules to govern its behavior in order for the multi-agent system to generate complex emergent behaviors during individual agent interactions. In certain multi-agent systems, the interactions also result in the formation of relationships, which can be leveraged for cooperative or collaborative activities. But these relationships often constrain individual-agent actions, since relationships imply that at least one contract between the agents must exist. In open multi-agent systems, where agents can come from anywhere, may be buggy or malicious, and run in dynamic, failure-prone environments, the contract is crucial in establishing a cooperative relationship. But the strength and stability of a relationship depends on how each agent satisfies the requirements in their mutual contracts. There is always some uncertainty as to whether or not either agent can or will satisfy some contract requirement – especially at the creation of a new contract. But in order to maintain the existence of a contract, each agent must overcome this uncertainty and assume that the other will do the same. The mechanism that facilitates this *act of faith* is generally regarded as *trust*.

Trust can help deal with uncertainty by reducing the complexity of expectations in arbitrary situations involving risk, vulnerability, and interdependence[4]. It is particularly useful when something cannot be gauged precisely with

* Dariusz G. Mikulski . Email: dariusz.mikulski@us.army.mil .

° Supported by ARO grant W91NF-05-1-0314, AFOSR grant FA9550-09-1-0278, and NSF grant ECCS-11280.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
22 MAR 2012	**Technical Report**	**22-02-2012 to 22-03-2012**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Trust Method for Multi-Agent Consensus	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
Dariusz Mikulski; Frank Lewis; Edward Gu; Greg Hudas	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Automation & Robotics Research Institute,The University of Texas at Arlington,7300 Jack Newell Blvd. S.,Fort Worth,TX,76118	**; #22693**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
U.S. Army TARDEC, 6501 East Eleven Mile Rd, Warren, Mi, 48397-5000	**TARDEC**
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
	#22693

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

For the 2012 SPIE Defense, Security and Sensing

14. ABSTRACT

The consensus problem in multi-agent systems often assumes that all agents are equally trustworthy to seek agreement. But for multi-agent military applications - particularly those that deal with sensor fusion or multi-robot formation control - this assumption may create the potential for compromised network security or poor cooperative performance. As such, we present a trust-based solution for the discrete-time multi-agent consensus problem and prove its asymptotic convergence in strongly connected digraphs. The novelty of the paper is a new trust algorithm called RoboTrust, which is used to calculate trustworthiness in agents using observations and statistical inferences from various historical perspectives. The performance of RoboTrust is evaluated within the trust-based consensus protocol under different conditions of tolerance and confirmation.

15. SUBJECT TERMS

Trust Model, Multi-Agent Systems, Consensus, Convergence

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Public Release**	**14**	

reasonable time or effort. The benefits of trustworthy relationships include lower defensive monitoring of others, improved cooperation, improved information sharing, and lower levels of conflict[5]. But the reliance on trust also exposes people to vulnerabilities associated with betrayal, since the motivation for trust – the need to believe that things will behave consistently – exposes individuals to potentially undesirable outcomes. Thus, trust is a concept that must not only be managed, but also justified.

In this paper, we examine trust within the context of the consensus problem. The literature often assumes that all agents in cooperative teams are equally trustworthy to seek agreement to resolve a consensus problem. But for multi-agent military applications – particularly those that deal with sensor fusion or multi-robot formation control – this assumption may create the potential for compromised network security or poor cooperative performance. The literature contains extensive work on consensus protocols[6,7,8] – but only a handful of researchers have attempted to incorporate trust into these consensus protocols[9]. Trust is typically factored in as a weight on some edge between agents in a communications digraph. And a consensus protocol utilizing the trust weights causes agents to converge towards values held by the most trustworthy agents (i.e. agents with incoming edge weights that are generally high). The devil in the details, however, is the manner on which the protocol assigns trust weights and how those trust weights change over time.

This paper begins by introducing the consensus problem and providing a distributed, discrete-time, trust-based protocol to this problem. Specifically, we prove that our protocol asymptotically converges to an agreement space in strongly-connected digraphs. Afterwards, we present the RoboTrust algorithm, a novel dynamic trust model that can meaningfully cultivate trust in multi-agent systems. Finally, we incorporate RoboTrust into our trust-based consensus protocol and analyze its performance in a simple three-agent network.

2. OVERVIEW OF THE CONSENSUS PROBLEM

In multi-agent systems, consensus means that each agent reaches an agreement with all other agents in the network about a particular quantity of interest. This is generally done through a consensus protocol, which defines how agents will interact with each other in order to update their current state. It has been seen that the consensus problem has broad applications to multi-agent systems in areas such as cooperative control, formation control, flocking, and sensor fusion.

In this section, we present a high-level overview of the consensus problem. Then, we provide a distributed, discrete-time, trust-based consensus protocol and prove its asymptotic convergence to an agreement space.

2.1 Consensus Problem Statement

Let $x_i \in \mathbb{R}^n$ be the public decision vector for each agent $i \in N$. Consider a network of decision-making agents with dynamics $\dot{x}_i = u_i$ interested in reaching a consensus via local communication with their neighbors on a graph $G = (N, E)$, where N is the set of all agents and $E \subseteq N \times N$ is the set of all directed edges between the agents. Let N_i be the set of first-neighbors of agent i. Consensus, by definition, means that all agents asymptotically converge to a one-dimensional agreement space that is characterized by:

$$x_1 = x_2 = \cdots = x_{|N|} \tag{1}$$

In other words, the solution can be described as a vector $x = \alpha\mathbf{1}$, where $\mathbf{1} = (1, 1, \cdots, 1)^T$ and α is a scalar real value equal to the final consensus value. It has been shown in the literature that $\dot{x} = -Lx$ is a distributed consensus algorithm that solves the consensus problem. Here, L is the Graph Laplacian defined as the difference between the diagonal degree matrix (D) and the adjacency matrix (A). The elements of A, denoted as A_{ij}, represent the number of directed edges from i to j (i.e. $A_{ij} \in \{0, 1\}$) and the diagonal elements of D, denoted as D_{ii}, are the number of outgoing edges incident to i. Note that $D_{ij} = 0$ for all $i \neq j$.

2.2 Distributed, Discrete-Time, Trust-Based Consensus Protocol

In order to develop any trust-based method, we must first define how to manage trust between each agent in a system. In our methods, we use a $|N| \times |N|$ matrix T that conforms to the following definition:

$$T = \left[t_{i,j}\right]_{|N|\times|N|} = \begin{cases} t_{i,j} = 1, & i = j \\ t_{i,j} \in [0,1], & i \neq j \end{cases} \tag{2}$$

This matrix is populated with values $t_{i,j}$ that represent the probability that agent j is trustworthy from the perspective of agent i. The values $t_{i,j}$ can also be interpreted as the probabilities that agent i will allow agent j to interact with him, since rational agents would prefer to interact with more trustworthy agents. Note that the diagonal of the matrix consists of fixed 1 values, indicating that each agent fully trust himself.

We now present the following trust-based consensus protocol for discrete-time agents with dynamics $x_i(k+1) = x_i(k) + \epsilon u_i(k)$ for a fixed graph topology

$$u_i(k) = \Delta_i^{-1}(k) \sum_{j \in N_i} T_{ij}(k) A_{ij}\big(x_j(k) - x_i(k)\big) \tag{3}$$

where $\Delta_i(k) = \sum_{j \in N} T_{ij}(k) A_{ij}$ is the weighted degree of all outgoing edges of i and $0 < \epsilon < 1$ is the step-size.

This protocol can be expressed in matrix form as:

$$u(k) = -(I - D^{-1}W)x(k) \tag{4}$$

where $W = T \circ A$ is the weighted adjacency matrix, such that T is the trust matrix with $0 \leq T_{ij} \leq 1$ for all $i, j \in N$, A is the adjacency matrix, and the operator $\circ$ is the Hadamard product; and D is the weighted degree matrix of the graph for all outgoing edges, such that $D_{ii} = \Delta_i$ and $D_{ij} = 0$ for all $i \neq j$. The matrix resulting from $I - D^{-1}W$ is a normalized Laplacian matrix $D^{-1}L = D^{-1}(D - W)$. Note how equation 4 takes the form of the distributed consensus algorithm $\dot{x} = -Lx$.

It is well-known that the *discrete-time* collective dynamics for the consensus problem can also be written as

$$x(k+1) = Px(k) \tag{5}$$

where $P = I - \epsilon L$ and $\epsilon > 0$. Here, P is known as the Perron matrix of graph G with parameter ϵ. If we substitute the normalized Laplacian for L in P, then the collective dynamics of the network under our algorithm are

$$x(k+1) = [(1-\epsilon)I + \epsilon D^{-1}W]x(k) \tag{6}$$

We now present certain results that are well-known in the consensus literature. These are included mainly for the benefit of the reader to understand how we can claim the asymptotic convergence of protocol 3.

Lemma 1: Let G be a digraph with $|N|$ nodes. Then the Perron matrix P with parameter $0 < \epsilon < 1$ is a row-stochastic, non-negative matrix with a trivial eigenvalue of 1.

Since $P = I - \epsilon D^{-1}L$, we get $P\mathbf{1} = \mathbf{1} - \epsilon D^{-1}L\mathbf{1} = \mathbf{1}$, which means the row sums of P are 1. Thus, P is row-stochastic and has 1 as a trivial eigenvalue of P for all graphs since zero is an eigenvalue of L associated with the eigenvector $\mathbf{1}$. Furthermore, we notice that, by definition, the weighted adjacency matrix W is a non-negative matrix. Thus, $\epsilon D^{-1}W$ is also non-negative. Also, $(1-\epsilon)I$ is always non-negative for $0 < \epsilon < 1$. Since the sum of two non-negative matrices is a non-negative matrix, P is a non-negative matrix.

Lemma 2: Let G be a digraph with $|N|$ nodes. If G is strongly connected, then P is a primitive matrix with parameter $0 < \epsilon < 1$.

An irreducible stochastic matrix is primitive when it has only one eigenvalue with maximum modulus. Since G is strongly connected, then P is irreducible[10] and by Lemma 1, P is also stochastic. And according to the Perron-Frobenius theorem, the fact that P has an eigenvalue of 1 with a positive eigenvector $\mathbf{1}$ implies that this eigenvalue is the Perron root $\hat{\lambda} = 1$. Hence, any other modulus of eigenvalues of P must be strictly smaller than the Perron root (i.e. $|\lambda| < \hat{\lambda}$ for all eigenvalues λ of P, such that $\lambda \neq \hat{\lambda}$). Thus, P is a primitive matrix.

Theorem (Convergence): Consider a network of agents $x_i(k+1) = x_i(k) + \epsilon u_i(k)$ for a fixed, strongly connected graph G applying the distributed consensus protocol 3. Then, protocol 3 asymptotically solves a consensus problem.

Considering that $x(k) = P^k x(0)$, consensus is reached in discrete-time if the limit $\lim_{k\to\infty} P^k$ exists. According to the Perron-Frobenius theorem, this limit exists for primitive matrices and according to Lemma 2, P is a primitive matrix under the conditions of protocol 3. Thus, protocol 3 asymptotically solves the consensus problem.

3. ROBOTRUST – A DYNAMIC, TRUST MODEL

In this section, we propose the RoboTrust algorithm – a mechanism to meaningfully cultivate trust towards other agents in multi-agent systems. It has been designed in a way that allows practitioners to easily integrate it into robotic applications. The proposed trust model establishes context through the use of acceptance functions and interprets context through confirmation and tolerance parameters. It also overcomes a particular limitation found in other trust models in the literature with respect to the *discount factor*. In addition to showing the derivation of the trust model, we also provide some practical modifications and extensions to the trust model in order to make it more flexible for real world applications.

3.1 Shortcoming of the Discount Factor

It is known that trust dynamics are highly context dependent[11]. As such, many trust models incorporate a parameter known as the discount factor to reflect this dynamism in the weighting of historical evidence[12,13,14]. The discount factor captures how much an agent would value past observations relative to the present. With a low discount factor, past behavior is forgotten quickly and the estimated trustworthiness reflects more on another agent's recent behavior. A high discount factor, on the other hand, considers the long term behavior of another agent. The discount factor parameter is often hand-tuned by a practitioner, but some researchers have developed methods to dynamically update the discount factor over time[15].

The discount factor however presents a serious issue for certain applications. Conventional wisdom about trust suggests that trust should generally take a long time to cultivate and a short time to destroy. With the discount factor, however, trust cultivation and destruction change at the same rate – even if the discount factor dynamically changes relative to the level of trust. This may be acceptable for applications such as e-commerce, where trust is generally based on the average consumer feedback[16]. But it may be problematic for cooperative applications within dynamic environments, where trust is more volatile due to complex inter-dependencies between agents. The RoboTrust model we propose in this section overcomes this limitation by allowing trust cultivation to change at different rates, which lines up better with a natural intuition of trust. Hence, the discount factor is ultimately not required in the RoboTrust model.

3.2 Acceptance Functions

The purpose of an acceptance function is to interpret a set of measurements and decide whether or not an agent should collectively deem them as acceptable. In essence, the acceptance function mathematically describes a particular context as some region in a feature space, and then defines which portions of that region the agent should find favorable.

In our work, we describe an acceptance function z with an output that represents either a favorable (1) or unfavorable (0) result based on a set of observations V, given by the information function $\rho: k \times F \to V$, where k is time step, F is the set of feature space attributes (or dimensions), and $V = \{v | k \in \mathbb{N}, \forall f \in F: v \in \rho(k, f)\}$.

$$z(k): V(k, F) \to \{0,1\} \tag{7}$$

Note that the explicit mapping of the feature space to favorable/unfavorable regions is application-specific, and should be defined by the practitioner.

The primary motivation for describing the acceptance function's output as Boolean is that the meaning of the output is readily understood and does not require any interpretation beyond recognizing the Boolean value. Technically, an acceptance function could also be defined on some continuous range between unfavorable and favorable extremes – but it would require that the function knows how to interpolate between each extreme, which can be complicated and somewhat ambiguous in interpretation.

3.3 Derivation of the RoboTrust Model

This subsection presents the mathematical derivation of the RoboTrust model. Let us assume agent i acquires a sequence of observations from an acceptance function regarding agent j

$$z_{ij} = \{z_{ij}(0), z_{ij}(1), \cdots, z_{ij}(r)\} \tag{8}$$

where $z_{ij}(r)$ is the most recent observation. Let Z be a random variable associated with these observations with a discrete probability distribution p that depends on a parameter θ. Then, the likelihood function can be defined as:

$$\mathcal{L}(\theta|z_{ij}) = p_\theta(z_{ij}) = P_\theta(Z = z_{ij}) \tag{9}$$

Suppose agent i considers only the $c+1$ most recent observations of agent j. We refer to $c \in \mathbb{N}$ as the *confirmation* parameter. Then:

$$z_{ij}^c = \{z_{ij}(r-c), z_{ij}(r-c+1), \cdots, z_{ij}(r-1), z_{ij}(r)\} \qquad 0 \le c \le r \tag{10}$$

Note that $z_{ij}^0 = \{z_{ij}(r)\}$ and $z_{ij}^r = z_{ij}$. If we let the parameter θ be the probability that agent j will be observed favorably by agent i, then the trust attitude that agent i has towards agent j can be modeled by

$$T_{ij} = \min \begin{pmatrix} \arg\max \mathcal{L}(\theta_0|z_{ij}^0) \\ \arg\max \mathcal{L}(\theta_1|z_{ij}^1) \\ \vdots \\ \arg\max \mathcal{L}(\theta_c|z_{ij}^c) \end{pmatrix} \tag{11}$$

That is, we assign the trust attitude T_{ij} to equal the smallest, most likely probability taken from various historical perspectives. Since we assume that the acceptance function output is Boolean, let us suppose that each observation is a random variable that comes from a binomial distribution, where there are $\kappa = \sum_{q=0}^{c} z_{ij}(r-q)$ favorable observations from a total of $\eta = c+1$ most recent observations. Then the likelihood function can be defined as

$$\mathcal{L}(\theta|\kappa,\eta) = \binom{\eta}{\kappa} \theta^\kappa (1-\theta)^{\eta-\kappa} \tag{12}$$

We wish to find the maximum likelihood estimate for θ given κ and η. This can be done by setting the derivative of the log-likelihood to zero and solving for θ.

$$\log \mathcal{L}(\theta|\kappa,\eta) = \log \binom{\eta}{\kappa} + \kappa \log \theta + (\eta - \kappa) \log(1-\theta) \tag{13}$$

$$\frac{d}{d\theta} \log \mathcal{L}(\theta|\kappa,\eta) = \frac{\kappa}{\theta} - \frac{(\eta-\kappa)}{(1-\theta)} = 0 \tag{14}$$

$$\hat{\theta} = \frac{\kappa}{\eta} \tag{15}$$

$\hat{\theta}$ denotes the maximum likelihood estimate, which we can now use to calculate a trust attitude T_{ij} for agent i towards agent j.

$$T_{ij} = \min \begin{pmatrix} \frac{\sum z_{ij}^0}{1} \\ \frac{\sum z_{ij}^1}{2} \\ \vdots \\ \frac{\sum z_{ij}^c}{c+1} \end{pmatrix} \tag{16}$$

With this trust model, a single unfavorable observation will reduce the trust to zero. However, $c+1$ consecutive favorable observations will increase the trust to one. Hence, we see that the confirmation parameter c describes the number of consecutive favorable observations necessary before the next consecutive favorable observation results in complete trust (i.e. $T_{ij}=1$).

It is readily clear that from a practical perspective, the RoboTrust trust model in 16 is too restrictive to handle the wide variety of contexts that agents may be interested in. A practitioner may want to have more control over how unfavorable observations affect the trust attitude. Hence, we modify the RoboTrust model by adding an additional parameter known as the *tolerance* parameter $\tau \in \mathbb{N}$, where $0 \leq \tau \leq c$.

$$T_{ij} = \min \begin{pmatrix} \arg\max \boldsymbol{\mathcal{L}}\left(\theta_{\tau} | z_{ij}^{\tau}\right) \\ \arg\max \boldsymbol{\mathcal{L}}\left(\theta_{\tau+1} | z_{ij}^{\tau+1}\right) \\ \vdots \\ \arg\max \boldsymbol{\mathcal{L}}\left(\theta_{c} | z_{ij}^{c}\right) \end{pmatrix} = \min \begin{pmatrix} \frac{\sum z_{ij}^{\tau}}{\tau+1} \\ \frac{\sum z_{ij}^{\tau+1}}{\tau+2} \\ \vdots \\ \frac{\sum z_{ij}^{c}}{c+1} \end{pmatrix} \tag{17}$$

The tolerance parameter controls how the trust model tolerates unfavorable observations. The higher the tolerance parameter, the less impact unfavorable observations have on the current trust value. This is because the tolerance parameter forces the model to take into account at least $\tau+1$ number of observations. Hence, we see that the tolerance parameter describes the number of consecutive unfavorable observations necessary before the next consecutive unfavorable observation results in no trust (i.e. $T_{ij}=0$).

Before concluding, it is important to note a small issue that arises when applying RoboTrust in practice: the question on how to handle initial trust cultivation when the total number of observations is less than $c+1$ (i.e. $r<c$). While there are several approaches that could resolve this issue, we favor a pessimistic approach that initializes the sequence of acceptance observations with $c+1$ unfavorable observations and offsets r to equal c. This approach allows agents to minimize exposure to trust-based vulnerabilities in initial interactions when trust has not yet been properly cultivated.

3.4 RoboTrust Extension for Indirect Trust Aggregation and Propagation

This subsection describes a method to give agents the ability to use RoboTrust to gauge trust about other agents who are not first-neighbors and cannot be directly observed. Let us begin by defining the mth-neighbors of i as a recursive set function.

$$J_i^m = J_i^{m-1} \cup N_{J_i^{m-1}} \qquad m \geq 1, J^0 = \{i\} \tag{18}$$

Thus, j is a mth-neighbor of i if $j \in J_i^m \backslash J_i^{m-1}$. Note, however, that this does not imply that i is also a mth-neighbor of j since the network is modeled as a digraph. Furthermore, it may not be able possible to indirectly gauge the trust of every mth neighbor in system since unidirectional relationships in the graph prevent cooperative interactions between any two agents. Therefore, this extension is limited to agents who are the mth-neighbors in bidirectional relationships with i (directly or indirectly) along with these mth-neighbors own first-neighbors.

As we saw in section 3.3, the RoboTrust algorithm used direct acceptance observations as its input to calculate trust. For this extension, we use indirectly-acquired acceptance observations from neighbors. These indirect observations can be thought of as *recommendations*. Thus, there is no need to change an agent's trust model for a particular context. Rather, an agent relies on the hidden (or private) acceptance functions of its mth-neighbors to determine the acceptance observations. Since there may be multiple hops and multiple paths between two agents in a network, the indirect trust extension is simply a rule for trust information propagation and aggregation. This rule can be stated with the following equation:

$$z_{ij}(k) = \left\lfloor \frac{2\sum_{q \in N_i} z_{qj}(k)}{|N_i|+1} \right\rfloor \tag{19}$$

Equation 19 essentially states that if more than half of the indirect acceptance observations at a particular time step are favorable, then agent i can consider his own acceptance observation of agent j to be favorable (i.e. $z_{ij}(k) = 1$); otherwise, his acceptance observation of agent j is unfavorable (i.e. $z_{ij}(k) = 0$). The purpose of the floor function and extra constants in the numerator and denominator is to prevent division-by-zero problems and extraneous logic statements in implementations of this extension.

To illustrate this concept, consider a network of 5 agents, as shown in Figure 1. Agents 1 is connected to agent 5 through agent 1's first-neighbors 2, 3, and, 4. Thus, agent 5 is a second-neighbor of agents 1. That said, agent 5 is not the second-neighbors of agent 1 since agent 5 does not have any directed edges to agents 2, 3, and 4.

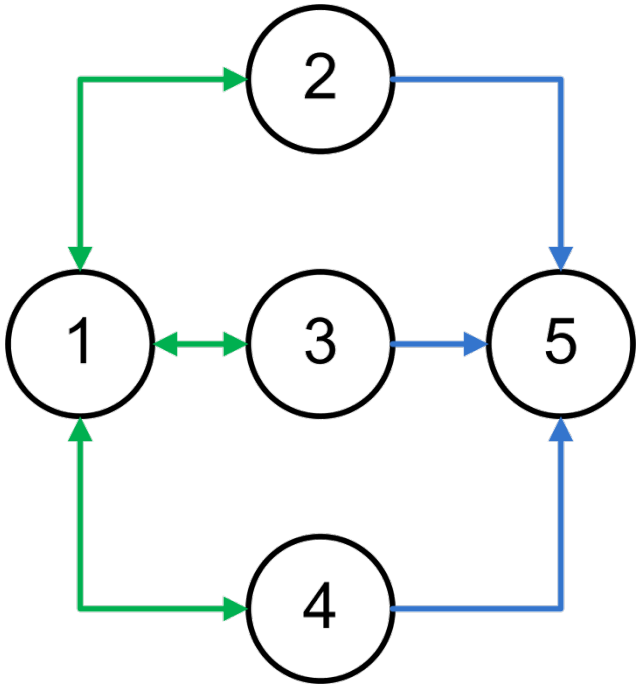

Figure 1. A 5-agent network where agent 5 is the second-neighbor of agent 1.

Suppose agent 1 is interested in indirectly gauging the trust of agent 5. Thus, agent 1 requests the latest 5 acceptance observations about agent 5 from its first-neighbors, namely agents 2, 3, and 4. Each one returns the following observation sequences about agent 5 to agent 1.

$$z_{2,5}^{4} = \{0 \quad 1 \quad 1 \quad 1 \quad 0\} \tag{20}$$
$$z_{3,5}^{4} = \{0 \quad 1 \quad 0 \quad 1 \quad 1\}$$
$$z_{4,5}^{4} = \{0 \quad 0 \quad 1 \quad 1 \quad 1\}$$

These sequences can be interpreted as recommendations about agent 5 from agents 2, 3, and 4. Agent 1 can now aggregate these observations using the equation in 19. In doing so, the indirect observation of agent 1 is

$$z_{1,5}^{4} = \left\{\left\lfloor\frac{2\times0}{3+1}\right\rfloor \quad \left\lfloor\frac{2\times2}{3+1}\right\rfloor \quad \left\lfloor\frac{2\times2}{3+1}\right\rfloor \quad \left\lfloor\frac{2\times3}{3+1}\right\rfloor \quad \left\lfloor\frac{2\times2}{3+1}\right\rfloor\right\} = \{0 \quad 1 \quad 1 \quad 1 \quad 1\} \tag{21}$$

Agent 1 can now take $z_{1,5}^{4}$ and use it in RoboTrust (with its own tolerance and confirmation parameters) as if it directly observed agent 5.

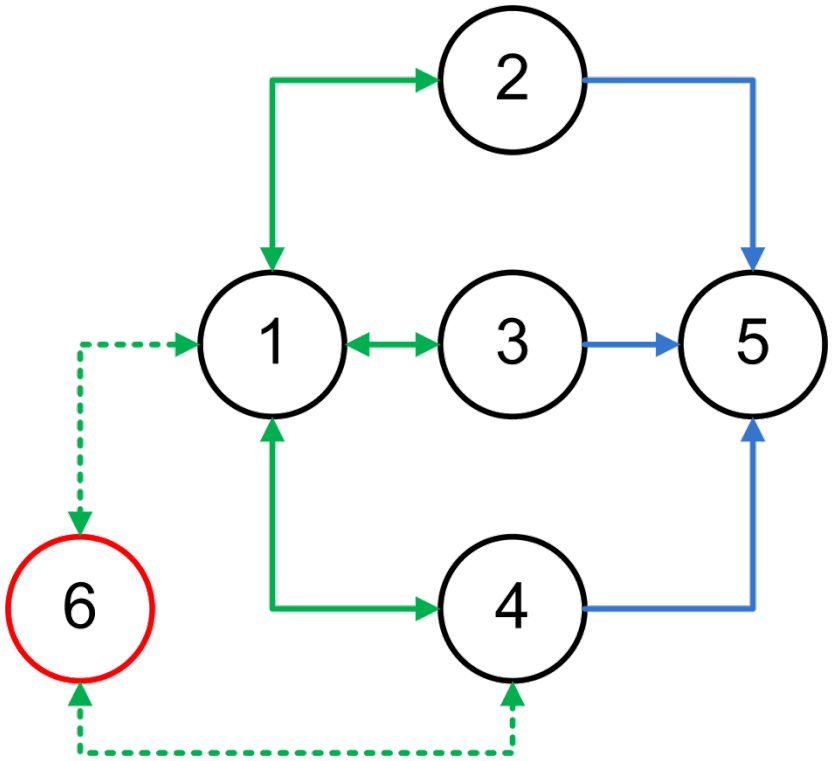

Figure 2. A 6-agent network where agent 5 is both the second-neighbor and third-neighbor of agent 6.

Equation 19 can also support trust propagation. Suppose agent 6 enters the network, becomes first-neighbors with agents 1 and 4 (as in Figure 2), and wants to indirectly gauge the trust of agent 5. Then, like before, agent 6 requests the latest 5 acceptance observations about node 5 from its first-neighbors.

$$z_{1,5}^{4} = \{0 \quad 1 \quad 1 \quad 1 \quad 1\} \tag{22}$$
$$z_{4,5}^{4} = \{0 \quad 0 \quad 1 \quad 1 \quad 1\}$$

Aggregating these observations with equation 19 results in

$$z_{6,5}^{4} = \left\{\left\lfloor\frac{2\times0}{2+1}\right\rfloor \quad \left\lfloor\frac{2\times1}{2+1}\right\rfloor \quad \left\lfloor\frac{2\times2}{2+1}\right\rfloor \quad \left\lfloor\frac{2\times2}{2+1}\right\rfloor \quad \left\lfloor\frac{2\times2}{2+1}\right\rfloor\right\}=\{0 \quad 0 \quad 1 \quad 1 \quad 1\} \tag{23}$$

Now, agent 6 can use $z_{6,5}^{4}$ in the RoboTrust algorithm to calculate the trust for agent 5. Thus, we see that trust information about agent 5 propagated to agent 6 from as far as its second-neighbors.

While the examples above illustrate ideal conditions, in practice, additional logic may be necessary to handle situations where agents are unresponsive or do not have acceptance observation data available. Trust values may also be used to filter the first-neighbors into a subset of trusted first-neighbors. These are application-specific conditions and are beyond the scope of this paper.

4. PERFORMANCE ANALYSIS OF THE TRUST-BASED CONSENSUS ALGORITHM

In this section, we analyze the trust-based consensus algorithm under two overarching conditions: fixed-trust and dynamic trust using a simple three-agent network.

4.1 Trust-Based Consensus with Fixed-Trust

For our fixed-trust demonstration, we utilize a simple three-agent directed network as depicted in Figure 3. By inspection, it can be readily seen that Figure 3 is strongly connected, which is a necessary condition for the convergence

of protocol 3. Each directed edge has been initialized with a trust value T_{ij}, designating the level of trust agent i has for agent j. Note that each agent has a directed edge that loops back onto itself with trust value equal to 1 – this signifies that each agent completely trusts itself.

Our intent for this subsection is to establish a convergence result for the trust-based consensus protocol in 3 using the graph in Figure 3. We keep all trust values fixed to their initial values in Figure 3, and set $x(0) = [10,20,30]^T$ and $\epsilon = 0.1$. The simulation terminates when $\sum_{i,j\in N}|x_i - x_j| < 1^{-5}$. The results of this simulation are shown in Figure 4.

In general, the practitioner of a fixed-trust consensus protocol assumes that the trust values are both known and independent of the decision value. These assumptions may be particularly useful in sensor fusion applications of similar sensors, where each sensor is known to lose its calibration at a known rate. Trust values can be extrapolated from calibration degradation curves using an application-specific formula.

4.2 Trust-Based Consensus with Dynamic Trust

The use of dynamic-trust signifies that agents are interested in cultivating trust with respect to particular contexts *during* the consensus process. Thus, unlike in the fixed-trust case, agents do not need to assume any pre-determined trust values nor that the trust is independent of the decision value. For our analysis, we use the RoboTrust algorithm in 17 to generate trust updates during the consensus process.

In order to use RoboTrust, we must first define at least one acceptance function, which describes a particular context by mapping out favorable regions in a feature space. Our choice of an acceptance function is arbitrary. So, for the purpose of this paper, we consider the context of an agent's *willingness to cooperate* to reach agreement during consensus. We measure an agent's willingness to cooperate by evaluating whether a particular action demonstrates a willingness to shorten the distance δ between disagreements. More specifically, we say that agent i observes agent j favorably if agent j's current state vector is closer to agent i's previous state vector than agent j's previous state vector.

$$\delta_{ij}(k) = \left\|x_i^{k-1} - x_j^{k-1}\right\| - \left\|x_i^{k-1} - x_j^{k}\right\| \qquad i \neq j, k > 0 \tag{24}$$

$$z_{ij}(k) = \begin{cases} 1 & \delta_{ij}(k) > 0 \\ 0 & \delta_{ij}(k) < 0 \end{cases} \tag{25}$$

In the event that $\delta_{ij}(k) = 0$ (i.e. $x_j^{k-1} = x_j^k$), there is some ambiguity since it is not clear whether or not agent j intends to cooperate with agent i. Agent j may have chosen to stay put for different reasons, not all of which may indicate malicious intent or unwillingness to cooperate. For example, agent j may be unexpectedly isolated because he perceives that all of his first-neighbors are uncooperative (i.e. agent j has no trust for any of his first-neighbors in this context). Note that agent i need not necessarily be a first-neighbor of agent j since it may be the case that $A_{ji} = 0$ even if $T_{ji} > 0$.

Handling this ambiguity is somewhat arbitrary and may be dependent on the specific application. For example, a practitioner may choose to resolve the ambiguity by making the result equal to a Boolean random variable taken from some probability distribution. However, since we intend to study the RoboTrust algorithm in a deterministic manner, we simply assign the acceptance value as the result of a negation on the previous acceptance value.

$$z_{ij}(k) = \{0,1\} \backslash z_{ij}(k-1) \quad \delta_{ij}(k) = 0 \tag{26}$$

For our dynamic trust demonstration, we utilize a simple three-agent agent network depicted in Figure 5. Note that the graph in Figure 3 is structurally the same as Figure 5 – but the initial trust values between agents are set to zero. Hence, the trust matrix is equal to the identity matrix (i.e. $T = I$). The rationale behind this trust initialization is based on the assumption that no agent has cultivated any trust with any other agent before the start of a consensus simulation.

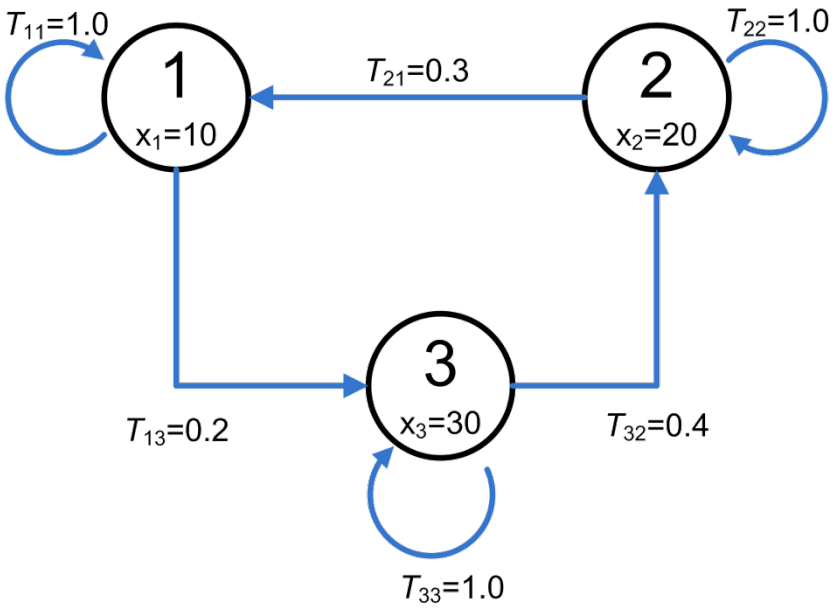

Figure 3. A simple, strongly connected three-agent network with initialized trust values.

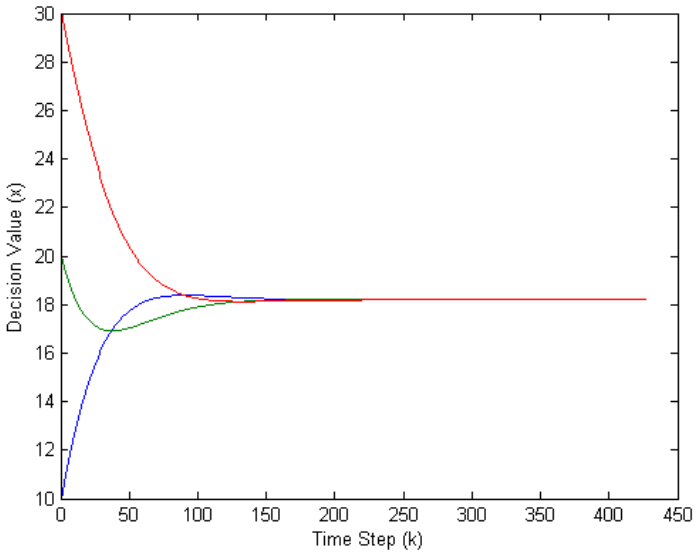

Figure 4. A consensus agreement is reached at value 18.1928 after 427 time steps. The initial trust values remained fixed throughout the entire simulation.

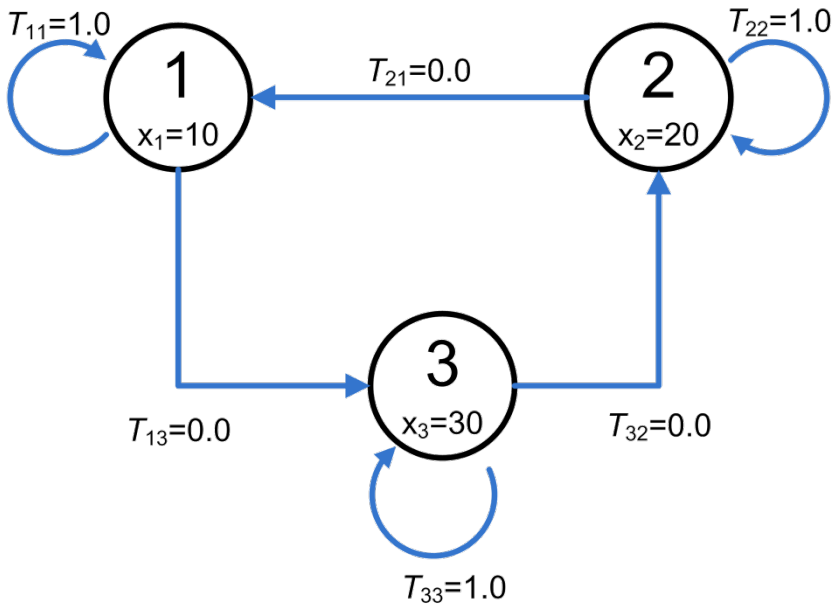

Figure 5. A simple, strongly connected three-agent network initialized with no trust between agents.

We begin by establishing a convergence result for the trust-based consensus protocol in 3 using the RoboTrust algorithm for trust dynamics. As before, we set $x(0) = [10,20,30]^T$ and $\epsilon = 0.1$, with the simulation termination condition $\sum_{i,j\in N}|x_i - x_j| < 1^{-5}$. Each agent uses the acceptance function in 25 and 26 to evaluate favorable or unfavorable observations of other agent behavior during the consensus process. These results are then placed into an appropriate acceptance observation sequence and used by the RoboTrust algorithm in 17 to provide a new trust value for the next time step. For this particular simulation, we set τ=3 and $c = 5$ for all agents. The results of this simulation are shown in Figure 6.

From a visual comparison, the time series profiles look markedly different between Figure 4 and Figure 6. In particular, we see time series sections in Figure 6 where $u_i(k) = 0$ *before* the final convergence result was reached. These sections represent portions of the convergence process when a particular agent i becomes isolated from his first-neighbors due to a total absence of trust in his first-neighbors. For example, at $k = 42$, the decision values of agent 1 and 3 intersect. At the next time step, agent 1 begins to lose some trust toward agent 3 because agent 3 appears to be increasing the disagreement distance. Eventually by $k = 46$, agent 1 loses all trust for agent 3 (its only first-neighbor) and flatlines till $k = 68$, which interestingly is an intersection point between agent 2 and agent 3. Figure 7 shows a detailed representation of the Figure 6 to serve as a reference for the above example.

To conclude our analysis, we seek to understand how the tolerance and confirmation parameters affect the number of time steps necessary to reach convergence. Hence, we executed 5150 simulations of our trust-based consensus protocol with RoboTrust using every (τ, c) pair within the ranges of $0 \leq \tau \leq 100$ and $1 \leq c \leq 100$. The simulation used the graph in Figure 5 with $x(0) = [10,20,30]^T$, $\epsilon = 0.1$, and the simulation termination condition $\sum_{i,j\in N}|x_i - x_j| < 1^{-5}$. As before, RoboTrust used the acceptance function in 25 and 26 to evaluate favorable or unfavorable observations of other agent behavior during the consensus process. The graphical results of these simulations are depicted in Figure 8.

In general, we see that higher tolerance values tend to shorten the length of time necessary to reach convergence. Also, higher confirmation values tend to extend the length of time necessary to reach convergence. There is however a notable exception – when $\tau = 0$, an increase in c does not significantly extend the length of time till convergence. We suspect that this exception may be due to how our acceptance function handles the boundary condition of $\delta_{ij}(k) = 0$. For this case, since our acceptance function is deterministic, it may end up oscillating between 0 and 1 due to the lack of tolerance, meaning that trust values will also oscillate between 0 and 0.5 – regardless of how large c is. This case highlights the importance of understanding the context of how trust is to be determined.

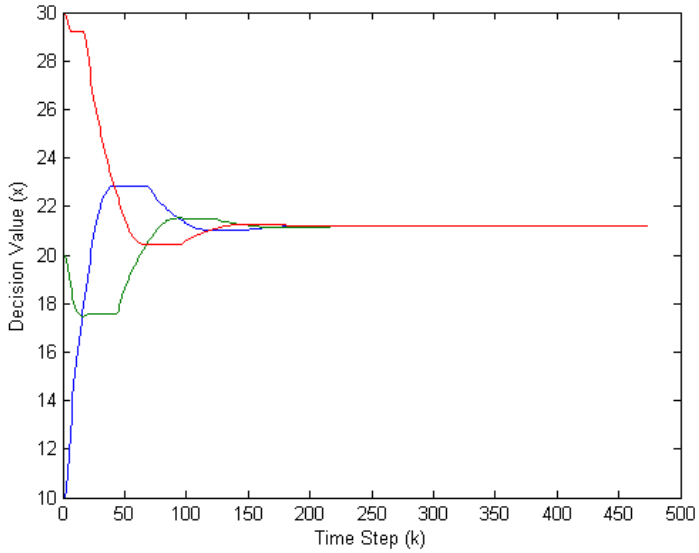

Figure 6. A consensus agreement is reached at value 21.1814after 473 time steps using the RoboTrust algorithm with parameters τ=3 and $c = 5$.

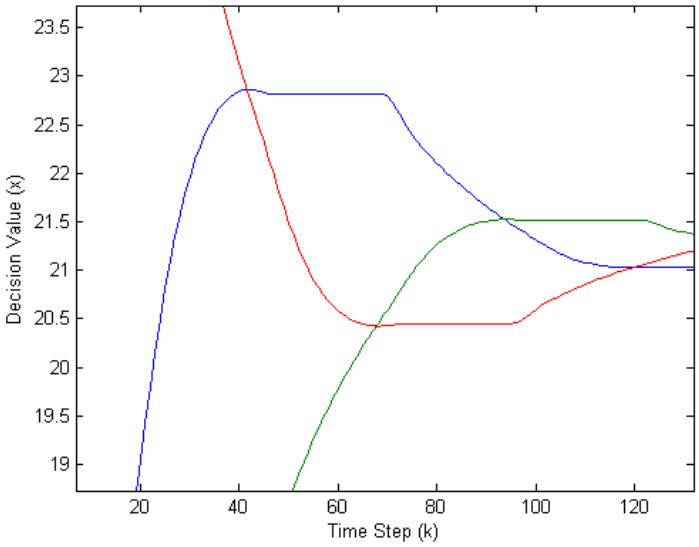

Figure 7. A closer and more detailed view of Figure 6. Agents 1 and 3 intersect at $k = 42$. Agents 2 and 3 intersect at $k = 68$.

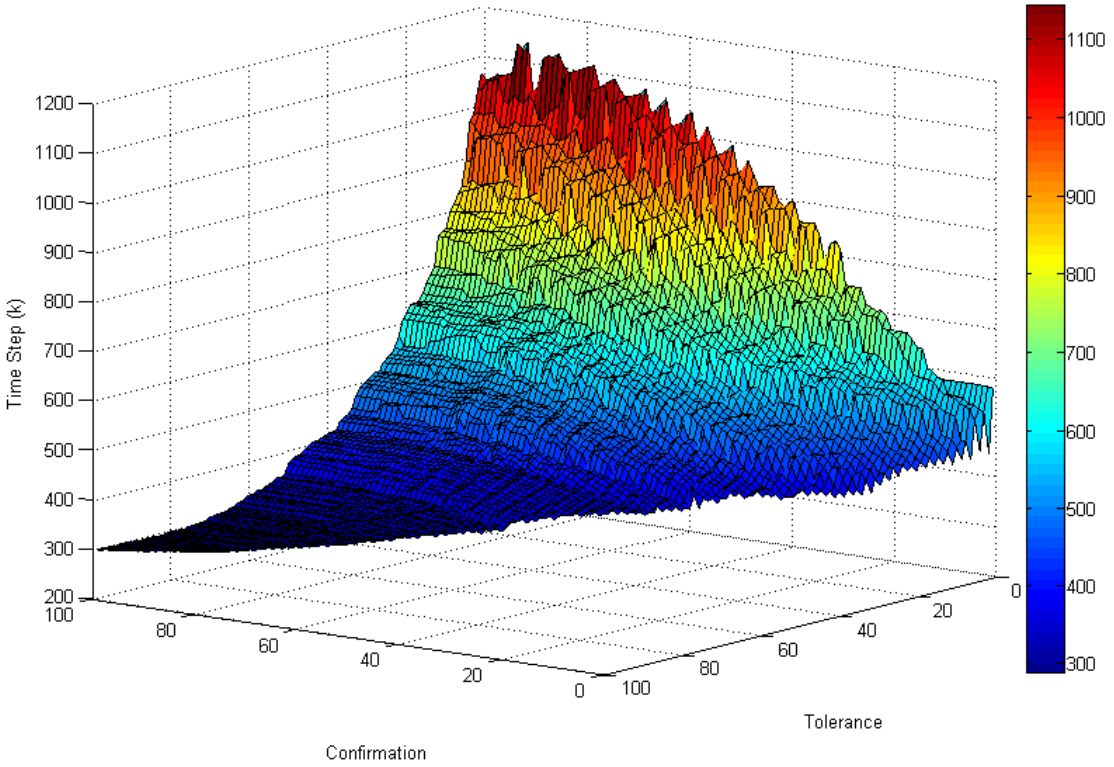

Figure 8. Surface plot of the number of time steps k needed for convergence on a (τ, c) pair for the digraph in Figure 5. The range represented in the plot is $0 \leq \tau \leq 100$ and $1 \leq c \leq 100$.

5. CONCLUSION

In this paper, we presented a discrete-time, trust-based consensus protocol and proved its asymptotic convergence to an agreement space for strongly connected digraphs. We also proposed a novel dynamic trust model, named RoboTrust, which finds the smallest, most likely probability from various historical perspectives of acceptance to gauge trust. Simulations were used to evaluate RoboTrust within the trust-based consensus protocol. In general, we learned that that higher tolerance values and lower confirmation values tend to shorten the length of time necessary to converge to consensus.

REFERENCES

[1] Singer, P.W., [Wired For War], Penguin Press, New York, NY, (2009).

[2] NATO Research and Technology Organization, "Multi-Robot Systems in Military Domains," NATO RTO Technical Report RTO-TR-IST-032, (2008).

[3] Chandler, P. and Pachter, M., [UAV Cooperative Decision and Control], SIAM, Philadelphia, PA, 15-35, (2009).

[4] Luhmann, N., "Familiarity, Confidence, Trust: Problems and Alternatives," In D. Gambetta (Ed.) [Trust: Making and Breaking Cooperative Relations], Basil Blackwell, New York, 94-108 (1988).

[5] Adams, B. & Webb, R., "Trust in Small Military Teams," 7th International Command and Control Technology Symposium, (2002).

[6] Ren, W., Beard, R. W., and Kingston, D. B., "Multi-agent Kalman consensus with relative uncertainty," Proc. Amer. Control Conf., 1865-1870 (2005).
[7] Olfati-Saber, R., Fax, J. A., and Murray, R. M., "Consensus and Cooperation in Networked Multi-Agent Systems," Proc. of the IEEE 95, (2007).
[8] Spanos, D. P., Olfati-Saber, R. and Murray, R. M., "Distributed Sensor Fusion using Dynamic Consensus," Proc. 16th IFAC World Congress, (2005).
[9] Ballal, P. and Lewis, F., "Trust-Based Collaborative Control for Teams in Communication Networks," Proc. 26th Army Science Conference, (2008).
[10] Meyer, C., [Matrix Analysis and Applied Linear Algebra], SIAM, Philadelphia, PA, 661–704 (2000).
[11] Mikulski, D., Lewis, F., Gu, E., and Hudas, "Trust Dynamics in Multi-Agent Coalition Formation," Proc. SPIE 8045, (2011).
[12] Wang, Y., & Vassileva, J., "Trust and reputation model in peer-to-peer networks," Proc. 3rd International Conference on Peer-to-Peer Computing (P2P), 150–157 (2003).
[13] Khosravifar, B., Gomrokchi, M., & Bentahar, J., "Maintenance-based trust for multiagent systems," Proc. 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 1017–1024 (2009).
[14] Mistry, O., Gürsel, A., and Sen, S., "Comparing trust mechanisms for monitoring aggregator nodes in sensor networks," Proc. 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 985–992 (2009).
[15] Wang, Y., Hang, C., and Singh, M. "A Probabilistic Approach for Maintaining Trust Based on Evidence," Journal of Artifical Intelligence Research 40, 221-267 (2011).
[16] Hazard, C.J., Singh, M.P., "Intertemporal Discount Factors as a Measure of Trustworthiness in Electronic Commerce," IEEE Transactions on Knowledge and Data Engineering 23(5), 699-712 (2011).